

Graph based KNN Variants for extracting Keywords

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the graph based KNN variants, and apply them to the keyword extraction. The initial KNN version was previously modified and applied to the keyword extraction which was mapped into a binary classification. In this research, we mention the three KNN variants, in case of the numerical vector-based versions: one where the selected nearest neighbors are discriminated by their similarities, one where the attributes are discriminated by their correlations with the categories, and one where the training examples are discriminated by their credits. In this research, we modify the three KNN variants into the graph-based versions, as the approaches to the keyword extraction, as well as the initial KNN version. This research is intended to improve the keyword extraction performance by modifying them so.

I. INTRODUCTION

The keyword extraction is the process of extracting important words as representative ones of a given text, selectively. It was interpreted into a binary word classification where each word is classified into keyword or non-keyword as a scheme of implementing a keyword extraction system. Because the keyword extraction is a domain dependent task where a same entity is classified differently depending on the given domain, it is regarded as the different task from the topic-based word classification where a word is classified into one or some among the predefined topics. In executing the keyword extraction, a text is indexed into a list of words, each word is classified into keyword or non-keyword by a machine learning algorithm, and words which are classified into keyword are extracted, externally. The exclusive keyword extraction is set as the coverage of this research, and it will be expanded into the fuzzy keyword extraction or the hierarchical keyword extraction in subsequent research.

This research is motivated by the successful application of the graph based KNN algorithm to the keyword extraction. An additional motivation for this research is the successful application of it to the topic-based word categorization. Words were encoded into graphs as the alternative representations to the numerical vectors, and the KNN algorithm was modified into the graph-based version. Its better performance was empirically validated in both tasks, compared with the traditional KNN algorithm. In this research, we derive the KNN variants and modify them into the graph-based versions as the approaches to the keyword extraction.

The idea of this research is to modify the three KNN variants into the graph-based versions which are used for extracting keywords from a text automatically. In the first variant, different weights are assigned to nearest neighbors by their similarities with or their distances from a novice example, in voting their labels. In the second variant, different weights are assigned to vertices or edges by their correlations with the output values in computing the similarities of a novice examples with the training examples. In the third variant, different weights are assigned to the training examples in computing their similarities with a novice item or voting nearest neighbors by their qualities. The three KNN variants are modified into the graph-based version for improving their performances in the keyword extraction.

Let us mention some benefits from this research. The problems such as the huge dimensionality, the sparse distribution, and the poor transparency, are solved by encoding words into graphs. The KNN variants substitute for the KNN algorithm for improving the classification performance. The KNN variants are modified into the graph-based versions by defining the similarity metric between graphs, for improving the classification performance, further. The better approaches are provided for implementing the keyword extraction system, as the goal of this research.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the KNN algorithm to its three variants. The directions of exploring previous works are derivation of KNN variants, modification of the standard KNN algorithm into its graph-based version as an approach to the keyword extraction, and the previous case of representing a text into a graph. The distinguishable points of this research are derivation of three KNN variants from the standard version, modification of them into their graph-based versions, and application of them to the keyword extraction. This article

is intended to explore previous works for providing the background for this research.

Let us explore the previous works on KNN variants. In 2001, the KNN variant where nearest neighbors are discriminated by their distances from or similarities as a novice item was applied to the text classification by Han et al. [1]. In 2008, MKNN (Modified K Nearest Neighbor) the KNN variants where training examples are discriminated by their validness, was proposed by Parvin et al. [4]. In 2018, the KNN variant where the attributes are discriminated by their correlations were proposed by Nababan and Sitompul [8]. However, this research uses different specific metrics for discriminating attributes, nearest neighbors, and training examples.

Let us explore the previous works on applying the modified version of KNN algorithm to the keyword extraction. In 2016, it was initially proposed that the graph-based version of KNN algorithm should be applied to the keyword extraction as a position paper by Jo [5]. In 2018, the modified KNN algorithm was validated in the task by Jo [7]. The journal article which describes in detail the modified KNN algorithm and its application to the keyword extraction is finally prepared for its publication by Jo [10]. This research is based on the three previous works.

Let us explore the previous works on representation of a text into a graph. In 2004, the process of representing a text into a graph was mentioned by Hensman [2]. In 2007, encoding a text into a graph for doing data mining tasks was covered by Jin and Srihari [3]. In 2020, the matching criteria of graphs which represent texts was proposed by Osman and Barukub [9]. The scheme of encoding a text into a graph and the similarity metric between graphs may be different in this research.

Let us mention the differences of this research from the previous works which were explored above. The KNN variants in the previous works are numerical vector-based versions, and they will be modified into graph-based versions in this research. As the expansion of [10], we modify and adopt the KNN variants for implementing the keyword extraction system. In this research, we connect representation of a text into a graph to the modification of the KNN variants. The modified KNN variants will be described in detail in Section III.

III. PROPOSED WORK

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a word into a graph and the proposed similarity metric. In Section III-B, we describe the standard version of KNN algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the keyword extraction system.

A. Similarity Metric

This section is concerned with the graph as a word representation and the similarity between graphs. In representing a word into a graph, a vertex indicates a text, and an edge indicates a similarity between texts. The similarity between edges is computed, before computing the similarity between graphs, viewing a graph as a set of edges. The similarity between graphs is computed by averaging over the similarities of all possible pairs of edges. In this section, we describe the process of encoding a word into a graph and the process of computing the similarity between graphs.

The process of encoding words into graphs is illustrated in Figure 1. In the vertex definition, from a text collection, texts which are relevant to the word are retrieved as the vertices. In the edge definition, the similarities among the retrieved texts are computed as edges. In the edge selection, the edges with their higher similarities are selected for representing a graph. Refer to [6] for studying the encoding process in more detail.

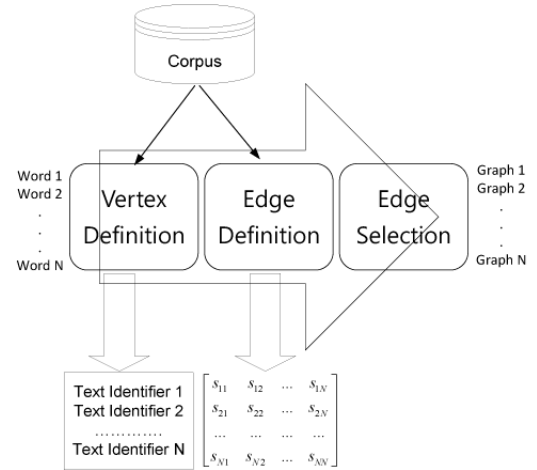


Figure 1. Process of Encoding Words into Graphs

The three cases of computing the similarity between two edges are illustrated in Figure 2. Each weight which is associated with its own edge is assumed to be a normalized value between zero and one, and if both nodes are identical to each other, the average over two weights is the similarity between two edges. If either of the two nodes is identical to each other, the product of two weights is the similarity between two edges. If neither of two nodes is identical, zero is the similarity between them. The similarity between two edges is always a normalized value between zero and one.

Let us mention the process of computing the similarity between graphs as a semantic similarity between words. The two graphs which represent words are notated by edge sets, $G_1 = \{e_{11}, e_{12}, \dots, e_{1|G_1|}\}$ and $G_2 = \{e_{21}, e_{22}, \dots, e_{2|G_2|}\}$. Two edges, $e_{1i} \in G_1$ and $e_{2j} \in G_2$, are associated with their own weights, w_{1i} and w_{2j} , and

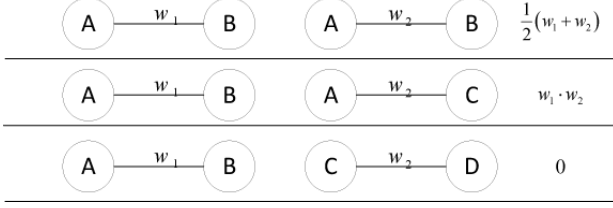


Figure 2. Three Cases of Comparing Two Edges with Each Other

the similarity between two edges, e_{1i} and e_{2j} by equation (1), (2), or (3), depending on the cases which are mentioned above,

$$\text{sim}(e_{1i}, e_{2j}) = \frac{1}{2}(w_{1i} + w_{2j}) \quad (1)$$

$$\text{sim}(e_{1i}, e_{2j}) = w_{1i} \times w_{2j} \quad (2)$$

$$\text{sim}(e_{1i}, e_{2j}) = 0 \quad (3)$$

The similarity between two graphs is computed by equation (4),

$$\text{sim}(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} \text{sim}(e_{1i}, e_{2j}). \quad (4)$$

The value which is generated from equation (4), is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(G_1, G_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding a word into a graph and computing the similarity between graphs. A word is encoded into a graph by the vertex definition, the edge definition, and the edge weighting. The three cases are considered for computing the similarity between edges. The similarity between graphs is computed by equation (4). In the future research, we consider representing a word into multiple graphs.

B. Initial Version of Graph based KNN Algorithm

This section is concerned with the initial version of graph based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 [6]. The similarity metric between graphs which was described in the previous section is used for computing the similarity between a novice graph and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into graphs, and they are notated by a set, $Tr = \{G_1, G_2, \dots, G_N\}$. A novice word is encoded into a graph

notated by G . Its similarities with the graphs which represent the sample words are computed by equation (4), as $\text{sim}(G, G_1), \text{sim}(G, G_2), \dots, \text{sim}(G, G_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

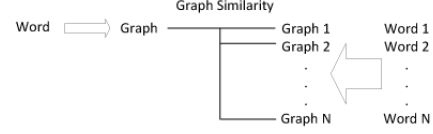


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, G), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

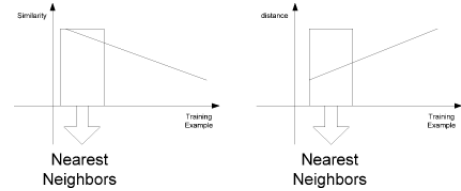


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(G_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, G , is decided by equation (8),

$$\text{label}(G) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the initial version of KNN algorithm from which we derive some variants. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with

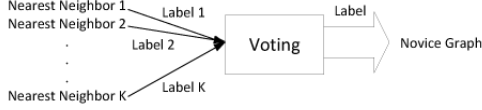


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. KNN Variants

This section is concerned with the variants which are derived from the KNN algorithm which was covered in Section III-B. The difference from the traditional version is to adopt similarity metric between graphs for computing the similarity between a novice item and a training example. In this section, we derive the three variants by discriminating nearest neighbors, edges, or training examples. The three variants of KNN algorithm are adopted for implementing the keyword extraction system. This section is intended to describe the three variants.

Let us mention the KNN variant which is derived by discriminating the nearest neighbors. A set of nearest neighbors is notated by $Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\}$, and its elements are assumed as $sim(G, G_1^{near}) \geq sim(G, G_2^{near}) \geq \dots \geq sim(G, G_k^{near})$. The weights, $w_1, w_2, \dots, w_k$, are assigned to the nearest neighbors, $G_1^{near}, G_2^{near}, \dots, G_k^{near}$, following, $w_1 \geq w_2 \geq \dots \geq w_k$, and the total weights are computed for the category, c_j by equation (9),

$$CS(Nr, c_j) = \sum_{G_i \in Nr} w_i \quad (9)$$

The label of the novice item, G , is decided by equation (10),

$$label(G) = \arg\max_{j=1}^m CS(Nr, c_j) \quad (10)$$

There are two ways of assigning weights to the nearest neighbors: exponentially decreasing way and arithmetic decreasing way as shown in Figure 6.

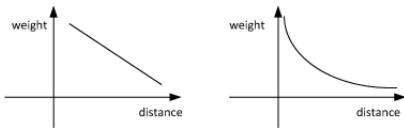


Figure 6. Discrimination over Nearest Neighbors

Let us mention the second graph based KNN variant where the edges are discriminated for computing the similarity between a novice item and a training examples as shown in Figure 7. The frequency of the edge, e_i , in the

words which are labeled with the category, c_j , is $f(e_i|c_j)$, and the total frequency of the edge, e_i , in the sample words, is $f(e_i)$. The influence of the edge, e_i , on the category, c_j , is computed by equation (11),

$$I(e_i, c_j) = \frac{f(e_i, c_j)}{f(e_i)}, \quad (11)$$

and the weight, w_i , is computed by equation (12),

$$w_i = \max_{k=1}^m I(e_i, c_j). \quad (12)$$

The equation (4) for computing the similarity between graphs is modified into equation (13),

$$sim(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} w_i w_j sim(e_{1i}, e_{2j}). \quad (13)$$

In this variant, equation (4) is replaced by equation (13) for computing the similarity between a novice item and a training example.

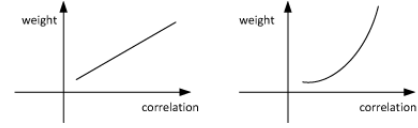


Figure 7. Discrimination over Edges

Let us mention the third KNN variant where the training examples are discriminated for computing the similarity between a novice item and a training example and/or voting the nearest neighbors for deciding the label as shown in Figure 8. The similarity threshold is used as the parameter, and for each training example, other training examples are taken within the threshold from it as its nearest neighbors. The number of nearest neighbors and the number of training examples which are labeled identically to the training example, G_i , are notated respectively by r_i and r_i^- , and the weight is computed by equation (14),

$$w_i = \frac{r_i^-}{r_i} \quad (14)$$

The similarity between the novice item, G , and the training example, G_i is computed by equation (15),

$$sim(G, G_i) = \frac{w_i}{|G| \times |G_i|} \sum_{j=1}^{|G|} \sum_{k=1}^{|G_i|} sim(e_j, e_{ik}). \quad (15)$$

and the total weight is computed for the category, c_j , by equation (16), in voting,

$$CS(Nr, c_j) = \sum_{G_i \in Nr} w_i. \quad (16)$$

Equation (15) and (16) are used respectively for computing the similarity between a novice item and a training example and voting the nearest neighbors for each category.

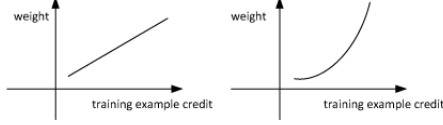


Figure 8. Discrimination over Training Examples

Let us make some remarks on the three variants of KNN algorithm which are proposed as the approaches to the keyword extraction. In the first variant, the nearest neighbors are discriminated for voting their labels. In the second variant, the edges are discriminated for computing the similarity between a novice input and a training example. In the third variant, the training examples are discriminated for voting the labels of the nearest neighbors and/or computing the similarity between graphs. We may consider the trainable KNN algorithm which optimizes the weights of the nearest neighbors, the attributes, and the training examples for minimizing the training error.

D. System Architecture

This section is concerned with the architecture and the execution flow of the keyword extraction system. In Section III-C, we described the three KNN variants which are adopted for implementing the keyword extraction system. The initial version of KNN algorithm which is mentioned in Section III-B is replaced by its variants in the architecture of the keyword extraction system which was previously proposed. The process of executing the system is to encode a word into a graph and classify it into keyword or non-keyword. This section is intended to describe the keyword extraction system which is implemented in this study.

The process of collecting the sample words and encoding them into string vectors for implementing the keyword extraction system is illustrated in Figure 9. The keyword extraction is viewed as a binary classification which classifies a word into keyword or non-keyword. The topic-based word classification belongs to the domain independent classification where a same word is always classified identically with regardless of the domain, whereas the keyword extraction is the domain dependent classification where a same word may be classified differently depending on the domain. The words which are labeled with keyword or non-keyword are collected domain by domain. It is required to tag the input text with its own domain for executing the keyword extraction.

The entire architecture of the proposed keyword extraction system illustrated in Figure 10. A text is given as the input and words are extracted from it in the indexing module. The sample words in the keyword group and the non-keyword group are encoded into graphs in the encoding module. The words which are indexed from a text are classified into one of the two categories in the similarity computation module and the voting module. The difference from the system

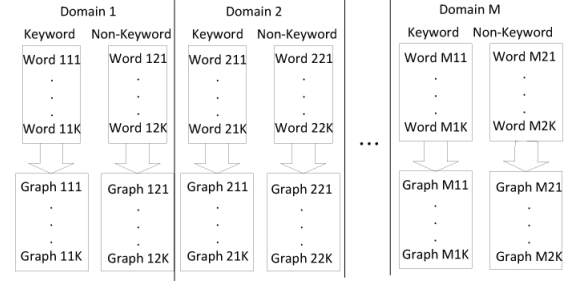


Figure 9. Preparation of Training Examples

architecture which is proposed in [7] is to replace the initial version of KNN algorithm by its variants.

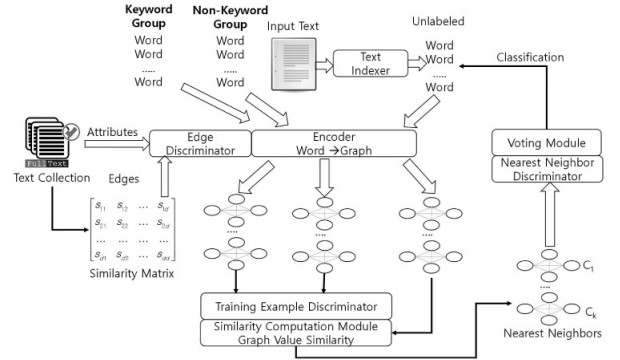


Figure 10. System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 11. The sample words which are labeled with keyword or non-keyword are collected in each domain, and they are encoded into graphs. An input text is indexed into a list of words, and they are also encoded into graphs. One of the three KNN variants is selected and applied to the classification of each word into keyword or non-keyword. The words which are classified into keyword are extracted as the final output.

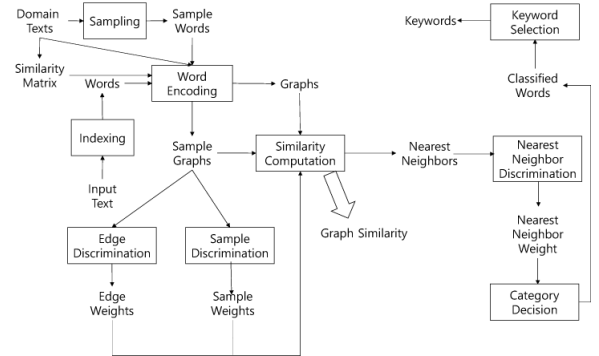


Figure 11. System Flow

Let us make some remarks on the proposed system which

is illustrated in Figure 10 as its architecture. The M domains are decided in advance, sample words each of which is labeled with keyword or non-keyword are prepared in each domain, and they are encoded into graphs. The words which are indexed from a text are classified by one of the three KNN variants which are described in Section III-C. The difference from the previous version of keyword extraction system is to provide the selection of the nearest neighbor discrimination, the edge discrimination, and the training example discrimination. We expect the better performance by replacing the initial version by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We encode words into graphs and apply the similarity metric among them to the KNN variants as well as the standard version. We derive the three variants from the standard version by discriminating the nearest neighbors, the edges, and the training examples. We design the keyword extraction system by adopt the KNN variants with the proposed similarity metric. In the next research, we will implement the keyword extraction system as a real system.

Let us mention the remaining tasks as the further discussions on this research. It is necessary to improve the speed of computing the similarity between graphs in the implementation level. Other machine learning algorithms such as the Naïve Bayes and the SVM (Support Vector Machine) are modified into the graph-based versions. The proposed versions of KNN variants are applied to other tasks such as the text classification and the text summarization. The keyword extraction system should be implemented by adopting the proposed approaches as a real program.

REFERENCES

- [1] E.H. Han, G. Karypis and V. Kumar, "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification" pp53-64, in *Advances in Knowledge Discovery and Data Mining. PAKDD 2001*, Berlin, Heidelberg:Springer, 2001.
- [2] S. Hensman, "Construction of conceptual graph representation of texts", *Proceedings of the Student Research Workshop at HLT-NAACL 2004*. 2004.
- [3] W. Jin and R. K. Srihari "Graph-based text representation and knowledge discovery", 807-811, *The Proceedings of ACM symposium on Applied computing* 2007.
- [4] H. Parvin, A. Hosein, and M.B. Behrouz, "MKNN: Modified k-nearest neighbor" *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1. Newswood Limited, 2008.
- [5] T. Jo, "Graph based KNN for Content based Word Classification", 24-29, *The Proceedings of 12th International Conference on Multimedia Information Technology and Applications*, 2016.
- [6] T. Jo, "Comparing Graph based K Nearest Neighbor with Traditional Version in Word Categorization in NewsPage.com, 12-18, *International Journal of Advanced Social Sciences*, 1, 2018.
- [7] T. Jo, "Graph based K Nearest Neighbors for Keyword Extraction in Current Affair Domain", 47-48, *The Proceedings of 1st International Conference on Advanced Engineering and ICT-Convergence*, 2018.
- [8] A.A. Nababan and O. S. Sitompul, "Attribute weighting based K-nearest neighbor using Gain Ratio", *Journal of Physics: Conference Series*, 1007, 1, 2018.
- [9] A.H. Osman and O.M. Barukub, "Graph-based text representation and matching: A review of the state of the art and future challenges", 87562-87583, *IEEE Access*, Vol 8, 2020.
- [10] T. Jo, "Modifying K Nearest Neighbor for Content based Word Classification by Graph Similarity Metric", in Preparation, 2023.